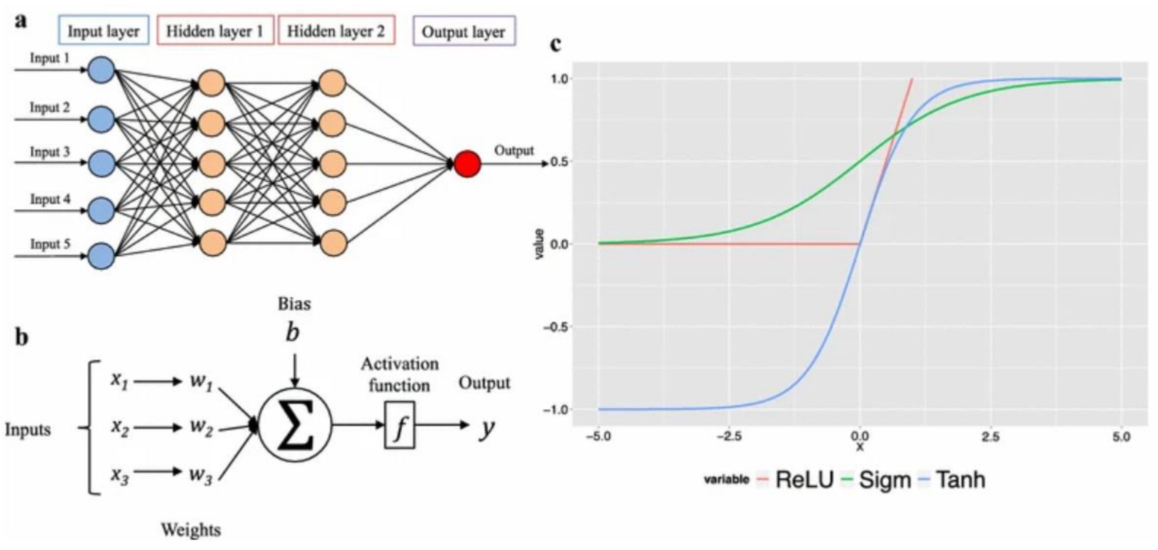
INTRODUCTION

Artificial intelligence and machine learning have become the most common techniques for handling data-related problems, and they are currently employed in a wide range of applications(Maglogiannis, 2007). ANN research has lately resurfaced, and is now categorized as deep-learning and has proven highly famous as a result of huge development of computational capacity, gaining great attention from both academic and business world(Chen et al., 2018).

To analyse vast amounts of sensory input, neural networks were developed to replicate the performance and efficiency of human brains influenced by the sophisticated cerebral cortex(Abiodun et al., 2018) . NNs seek to express non-linear and variable quantities through integrating numerous layers of representation via linked non-linear operations(Maglogiannis, 2007) . DL model are a subset of networked representation-learning methods that aim to extract and organise discriminatory features from input by identifying features that comprise multi-level feature representation(Bengio et al., 2013). With success, representation learning methods have been used to simulate complicated problems like image categorization, pattern recognition, and speech recognition

An NN is made up of fully connected layers: one input, hidden and output layer (Chen et al., 2018). One of the most common deep learning models is feed-forward NN, in which data goes without looping from the input via the hidden to the output layer, as shown in the picture below.

a) Fully connected 4 layered FFNN. b.) A neuron takes inputs and produces a value y. The output y is a combination of nonlinear input (with weights), passed through an activation function to generate nonlinearity. c Activation functions of popular neurons.



The usefulness of feedforward NN models in classification tasks has been established in several research and their effectiveness in the recognition of Handwritten digits has also been demonstrated in various experiments. In computer vision, handwritten digit recognition (HDR) is a critical subject(Albahli et al., 2020). Computerization of the postal system, processing of electronic bank cheques, writer identification, passport examination, and vehicle number plate recognition are just a few examples of the application of HDR(Savas & Eldén, 2007).

However, making use of the NN architecture might be difficult in certain cases because of the sheer volume of factors affecting its efficiency. (Al-behadili, 2016; Y. Lee, 1991). But by fine-tuning a few hyperparameters, the performance of the model may be enhanced. As a result, it is critical to choose the best hyperparameter values(Chugh et al., 2020).

A feedforward neural network is implemented in this project to classify digits using a standardized dataset from Sci-kit learn, and the performance is compared using a range of ML classifiers. By adjusting its hyperparameter, we will examine a variety of possibilities for building an ideal architecture.

BACKGROUND

Hand-written digits classification is frequently regarded as a typical problem as the assigning of an unknown object to one of the ten class labels is itself challenging(Cireşan et al., 2010). It usually results in a high variation of the objects within each class since objects from various classes may be similar. There are several methods to solve this problem, and various researchers have made substantial contributions.

In his article,(Savas & Eldén, 2007), employed two techniques using higher-order singular value decomposition to classify handwritten digits. The first approach employs HOSVD for building class models, while the subsequent procedure involves HOSVD for tensor estimate in two modes. Even though he attained a classification rate of 94%, the training time and computational resources required were huge and an attempt to vary non-self-adjusting network parameters was not done.

(Y. Lee, 1991) presented how Radial Basis Function (RBF) and KNN give equally good performance. However, findings from huge, high-input-dimensional data show that these classifiers are frequently constrained by practical limitations like training and testing time and memory utilization.

(Al-behadili, 2016)looked at the usage of artificial neural networks (ANN) to predict unknown handwritten digits, with an emphasis on binarization as a pre-processing method. Other typical machine learning models were examined but failed to analyze the optimization of their hyperparameters.

(Chugh et al., 2020)compare traditional ML models in image classification, such as the K-Nearest Neighbour, Multi-Layered Perceptron, and Random Forest classifier, in this study. MLP, RF, and KNN scored 89.57 %, 89.2 %, and 85.87 %, respectively. In addition, RF has the shortest computational time per 100 epochs, at 34.89

seconds, followed by KNN at 106.92 seconds, and MLP at 521.78 seconds. However, no attempt was made to modify these models or tune their hyperparameter values for digits recognition.

There is a scarcity of academic research that looks at the performance of a neural network when its hyperparameters are changed as most previous works use off-the-shelf ML models but failed to take hyperparameter optimization into account. Therefore, optimizing the hyperparameters of feedforward neural networks for classification (digit recognition) remains a knowledge gap that this study seeks to fill.

OBJECTIVE OF THE STUDY

Notwithstanding, the efficiency of an NN model, the determination of an optimal hyperparameter has a substantial influence on its effectiveness (Kim, 2020). Tuning of hyperparameter settings can substantially improve or impair the NN's overall performance (Zhang et al., 2020).

Furthermore, the number of distinct hyperparameter settings is vast. Finding the best NN hyperparameter settings involves considerable effort (Abiodun et al., 2018). As a result, hyperparameter optimization of a feedforward neural network is considered the main objective of this project with the primary aim of improving the baseline model.

METHODOLOGY

This section discusses the tools, dataset, neural network architecture, baseline classifiers, and methodology used to conduct the experiments.

Tools

The following up-to-date industry frameworks, packages, libraries, and tools, as well as IDE, were utilized to implement the digit classification using a feedforward neural network:

- Python 3.9 ,
- Keras
- TensorFlow
- Scikit-Learn
- Jupyter Notebook
- NumPy
- Seaborn
- Matplotlib

Dataset

To produce comparable findings and derive generalizable conclusions, the experiment is conducted on a benchmark and standard dataset that comes with sci-kit- learn, furthermore, it is a variant of the UCI ML manually written digits datasets test set

Images of handwritten digits are included in the data set: There are ten classes, each of which pertains to a handwritten digits in standardised bitmaps were extracted using NIST-provided pre-processing tools. It has an 8x8 input matrix with each member being an integer between 0 and 9. This decreases dimensionality and offers tiny distortions invariance("Dua, Dheeru and Graff, Casey, 2017).

Machine Learning Methods

The experiment was conducted in different settings, and the results were examined. We employed five ML models (Random Forest, Gaussian Naive Bayes SVM, KNN with Decision Trees) as baseline models to evaluate the neural network's effectiveness prior to and after their hyperparameters modification.

K-nearest neighbors (KNN)

We use KNN, as built in Scikit-learn. Just count of the nearest neighbours' hyperparameter were evaluated, while the others were left at their default values. The value of k was maintained constant at 5.

Naive Bayes

Naive-Bayes posits that the influence of a specific variable does not depend on the effect of other characteristics(Abiodun et al., 2018). Even if these traits are interrelated, they are nevertheless considered separately. Because this assumption simplifies computing, it is seen as naïve. This is known as class conditional independence (Lejeune, 2020). The hyperparameters were kept at their default value in sk-learn.

Decision Trees

DT is a graph that portrays the expected outcomes of a bunch of associated decisions. It allows an individual or association to analyze several options (A. Comeau & C. McDonald, 2019). A piecewise constant estimate is an example of a tree. (Albahli et al., 2020). The hyperparameters were kept at their default value in sk-learn.

Random Forest

A random forest consists of ensembled decision trees. Feature randomization and bagging are used to create every single tree with expectations of delivering a measurably independent forest whose committee prediction is more reliable than the individual trees.(Alpaydin et al., 2000; Bosch et al., 2007). The hyperparameters were kept at their default value in sk-learn.

Support Vector machine

The notion of decision planes, which establish decision limits, underpins SVM classification. A decision plane is a plane that splits a set of items with varying classes memberships. It identifies the vectors ("support vectors") that determine the separators with the greatest degree of separation of classes(Al-behadili, 2016; Alpaydin et al., 2000; Chen et al., 2018).

Feed Forward Neural Network

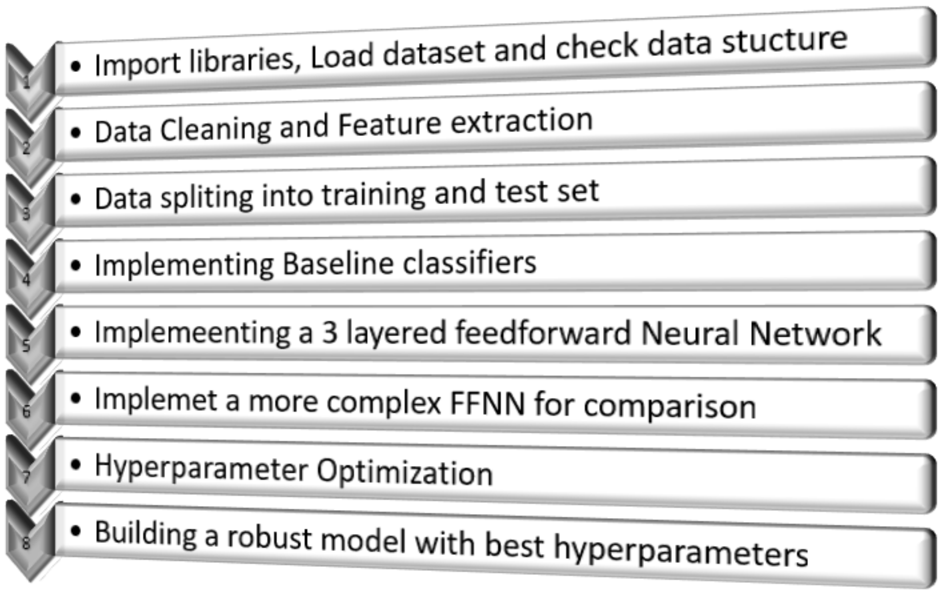
Handwritten digits are classified using a feedforward neural network with different architecture. Each layer has a specific number of nodes (the input layer has 64 nodes, each of the hidden layer, depending on the architecture, have 100 and the output ,10) also known as neuron. Nodes of one layer is linked with all nodes of the following layer. The determinant of the counts of nodes in the input layer is by the characteristics(features) of the data while that of the output is the apparent class of the dataset.

Grid Search Hyperparameter Optimization

Grid search is the most basic hyper - parameters optimization approach. Essentially, we partition the hyperparameter domain into a discrete grid. Then, utilizing cross-validation, we attempt every possible set of attributes from this grid, determining various performance measures. The optimal combination of settings for the hyperparameters is the point on the grid that maximises the average in cross-validation.

EXPERIMENTAL METHODOLOY

In carrying out this experiment, we followed the following steps.


1
• Import libraries, Load dataset and check data stucture
2
• Data Cleaning and Feature extraction
3
• Data spliting into training and test set
4
• Implementing Baseline classifiers
5
• Implemeenting a 3 layered feedforward Neural Network
6
• Implemet a more complex FFNN for comparison
7
• Hyperparameter Optimization
8
• Building a robust model with best hyperparameters

1. After importing libraries, we loaded the data and check the structure.

2. Decide how the data should be represented in the model. This entailed transforming features into numeric and expressing all data points as a vector of a predefined size. That is , flattening the image and merging it with other components.

3. Divide the data into two sets: testing and training. The train set is for creating the model, while the test set is for evaluating it

4. Train alternative models to understand how other algorithms perform in contrast to our feed-forward neural network.

5. Implement a three-layered NN model and fit it to the data.

6. Develop a more complex neural network to better understand their performance.

7. Tweak the algorithm's hyperparameters (one at a time and using a grid optimization approach) to find the best hyperparameter for improvement of he model's accuracy.

8. Build a robust network using the selected hyperparameters and assess its performance in comparison with the baseline algorithms using the evaluation metrics.

EXPERIMENTAL SETUP

Splitting the data into train and test set

We divided our data into 70 percent and 30 percent training and testing set respectively. When the testing and training split was done, the data was not shuffled. We did this to preserve the order of the data so that it could be returned for visualization.

Hyperparameters Tunned

Momentum, activation functions, learning rates, dropout, batch sizes, hidden layers, number of epochs, network weight, and model architectures are all examples of hyperparameters tunned in this experiment. The model architecture is made up of input, counts of hidden layers and nodes.

Optimization method

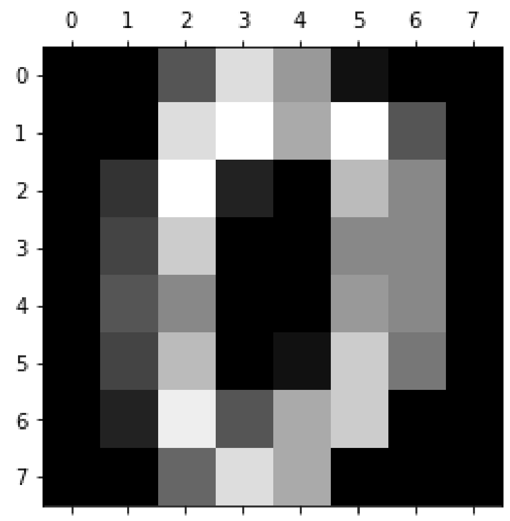
This study uses a grid searchCV approach to determine the best accurate classification model.

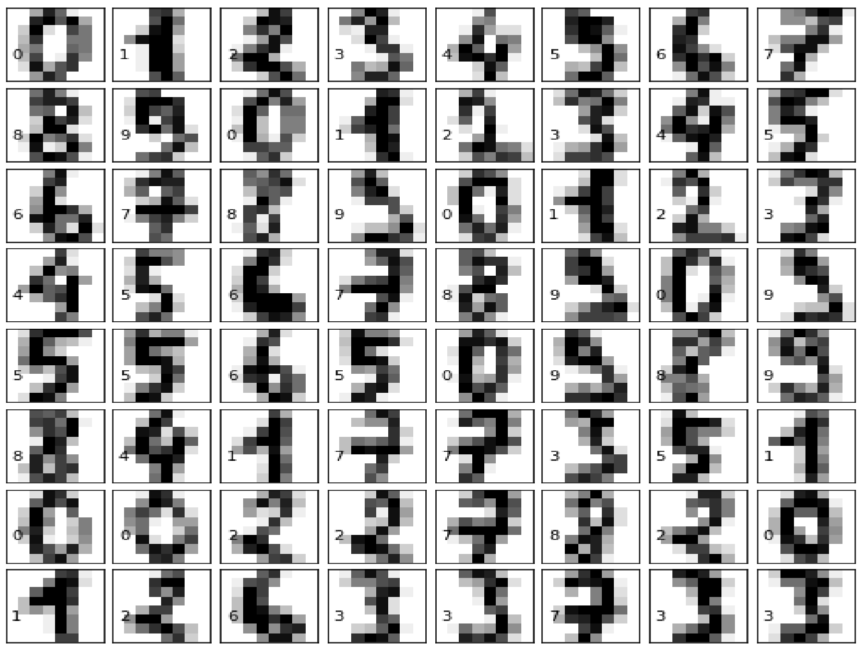
Evaluation metrics

To compare the result of the algorithms, this project used the training and validation accuracy and the computational time as the evaluation metrics.

Load and visualization of data

Scikit learn is used in loading the dataset, which is a database of 8x8 pictures (64 attributes) of numbers having 1797 sample points in total. The dataset is imported as digits.





Data pre-processing

In order to train a classifier,the two-dimensional arrays of the image data was flattened to a single dimension. We initialise and train a classifier after flattening and reshaping the data into a two-dimensional array including one 64 element vector for each image. The samples number (decided by calculating input size architecture) then the kneading of the inner structure defines the structure of the final data.NumPy infers the lengths of the row dimension from other dimensions in the structure when the second transform is set to -1.

Creating the baseline Classifiers

When working on a classification problem, it is common to evaluate various algorithms to assess how well they do in comparison with others (Al-behadili, 2016) .We decided to implement the other classifiers first before going into the feedforward neural network to give us a baseline for comparison of the FFNN. The following Sklearn classifiers are used:

- Support Vector Machine (SVM) : We used sklearn default hyperparameter with gamma = 0.0001

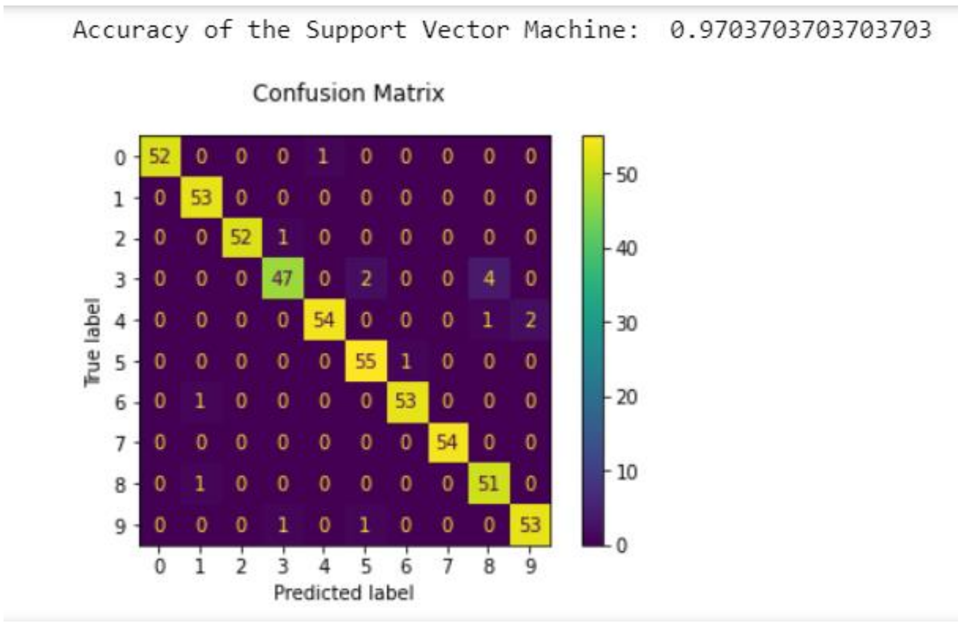
- Naive Bayes : We used sklearn default hyperparameter
- Decision Trees (DT) : We used sklearn hyperparameter
- Random Forest : Default sklearn hyperparameter was used
- K Nearest Neighbours (kNN) : We used sklearn hyperparameter with k = 5
- Neural Network : The model has three layers consisting of 64 input nodes and 100 nodes in in the hidden layer. While the output layer has 10 nodes. While ‘softmax was used for the output layer and activation function for the hidden layer was ‘tahn ‘. Number of epochs was 15 , batch size was 32 and 0.0 was dropout function. All other hyperparameters were left at their default values.

RESULTS

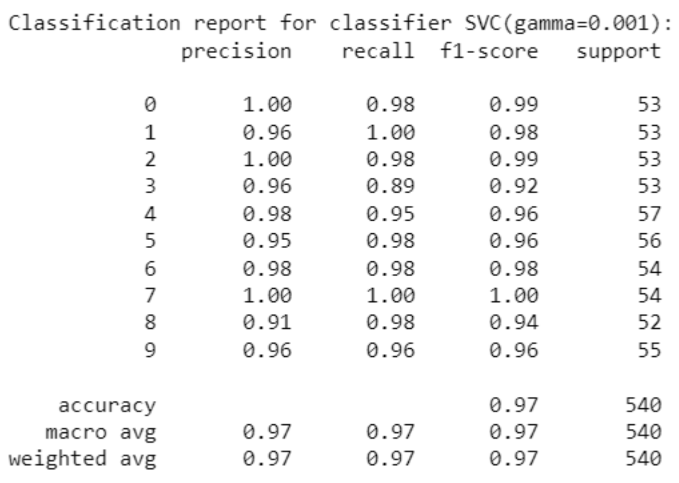
SUPPORT VECTOR MACHINE

SVM recorded a training accuracy of 99% with a test accuracy of 97%.

The confusion metrics below provide us a rapid and broad snapshot of our models' performance and the types of errors they make. The classification report gives us the f1-score, accuracy , precision and support.



```
Accuracy of the Support Vector Machine:  0.9703703703703703
```

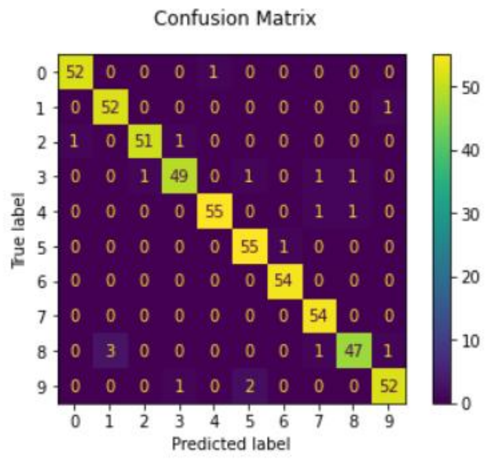


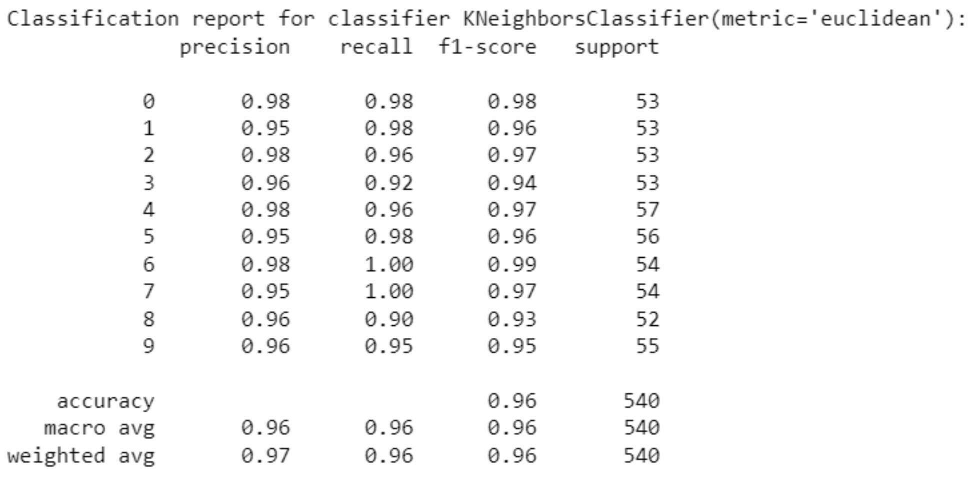
```
Classification report for classifier SVC(gamma=0.001):
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        53
           1       0.96      1.00      0.98        53
           2       1.00      0.98      0.99        53
           3       0.96      0.89      0.92        53
           4       0.98      0.95      0.96        57
           5       0.95      0.98      0.96        56
           6       0.98      0.98      0.98        54
           7       1.00      1.00      1.00        54
           8       0.91      0.98      0.94        52
           9       0.96      0.96      0.96        55

    accuracy                           0.97       540
   macro avg       0.97      0.97      0.97       540
weighted avg       0.97      0.97      0.97       540
```

K-nearest neighbours

KNN recorded a training accuracy of 99% with a test accuracy of 96%. With a training time of 0.003

```
Accuracy of the Algorithm:  0.9648148148148148
```

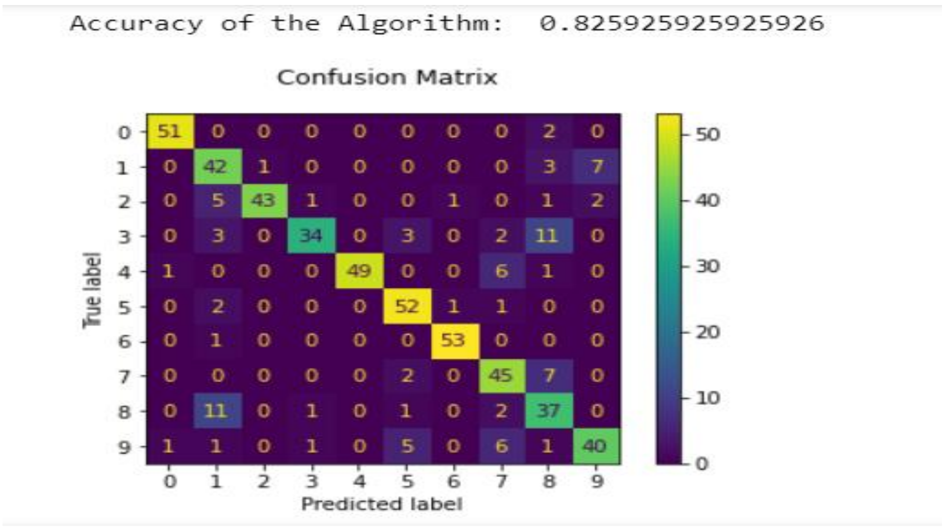




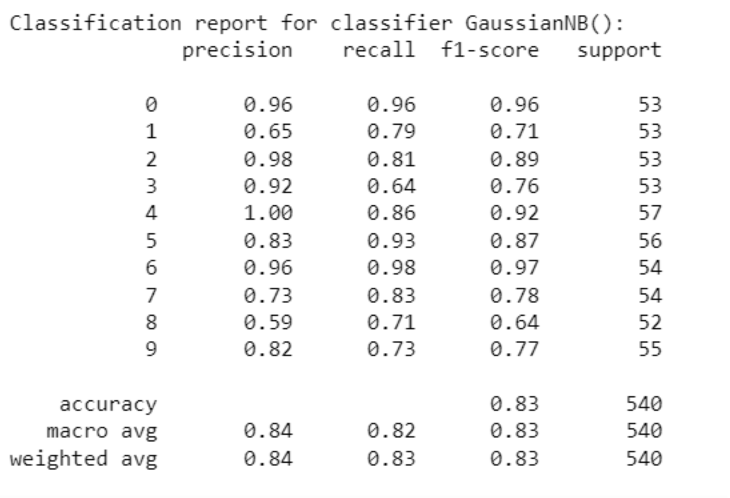
```
Classification report for classifier KNeighborsClassifier(metric='euclidean'):
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        53
           1       0.95      0.98      0.96        53
           2       0.98      0.96      0.97        53
           3       0.96      0.92      0.94        53
           4       0.98      0.96      0.97        57
           5       0.95      0.98      0.96        56
           6       0.98      1.00      0.99        54
           7       0.95      1.00      0.97        54
           8       0.96      0.90      0.93        52
           9       0.96      0.95      0.95        55

    accuracy                           0.96       540
   macro avg       0.96      0.96      0.96       540
weighted avg       0.97      0.96      0.96       540
```

GAUSSIAN NAÏVE BAYES

Gaussian Naïve Bayes recorded a training accuracy of 87% with a test accuracy of 82%. With a training time of 0.006.



```
Accuracy of the Algorithm:  0.825925925925926
```

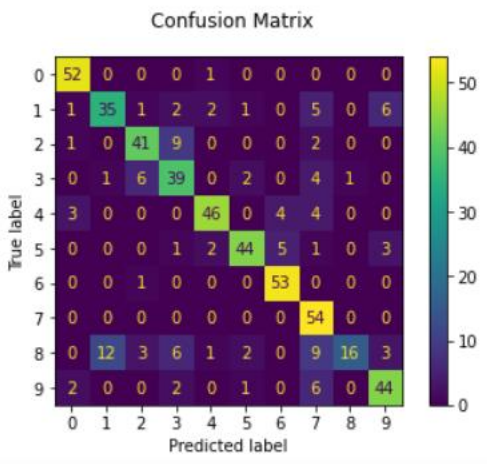


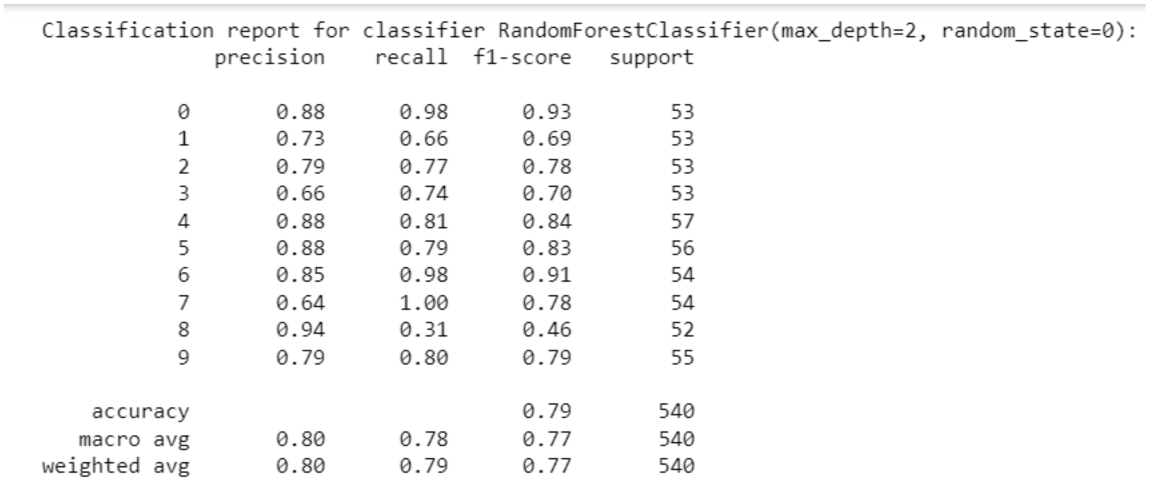
```
Classification report for classifier GaussianNB():
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        53
           1       0.65      0.79      0.71        53
           2       0.98      0.81      0.89        53
           3       0.92      0.64      0.76        53
           4       1.00      0.86      0.92        57
           5       0.83      0.93      0.87        56
           6       0.96      0.98      0.97        54
           7       0.73      0.83      0.78        54
           8       0.59      0.71      0.64        52
           9       0.82      0.73      0.77        55

    accuracy                           0.83       540
   macro avg       0.84      0.82      0.83       540
weighted avg       0.84      0.83      0.83       540
```

RANDOM FOREST CLASSIFIER

Random forest recorded a training accuracy of 84% with a test accuracy of 78%. With a training time of 0.00.

```
Accuracy of the Algorithm:  0.7851851851851852
```

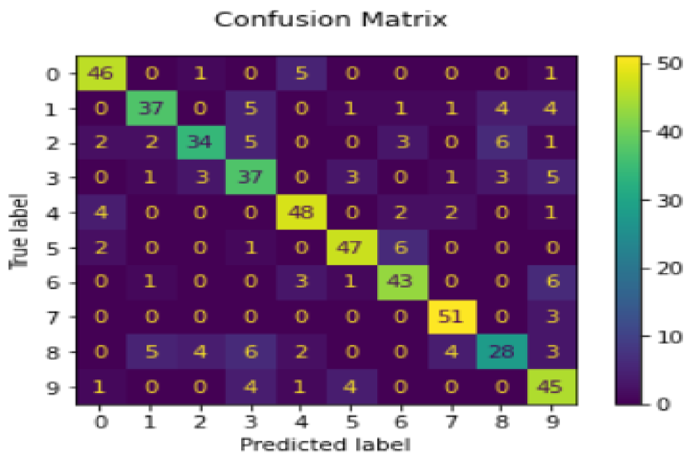




```
Classification report for classifier RandomForestClassifier(max_depth=2, random_state=0):
              precision    recall  f1-score   support

           0       0.88      0.98      0.93        53
           1       0.73      0.66      0.69        53
           2       0.79      0.77      0.78        53
           3       0.66      0.74      0.70        53
           4       0.88      0.81      0.84        57
           5       0.88      0.79      0.83        56
           6       0.85      0.98      0.91        54
           7       0.64      1.00      0.78        54
           8       0.94      0.31      0.46        52
           9       0.79      0.80      0.79        55

    accuracy                           0.79       540
   macro avg       0.80      0.78      0.77       540
weighted avg       0.80      0.79      0.77       540
```

DECISION TREES

Decision Trees recorded a training accuracy of 100% with a test accuracy of 77%. With a training time of 0.00.

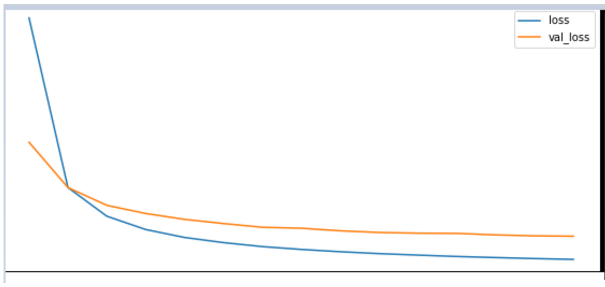


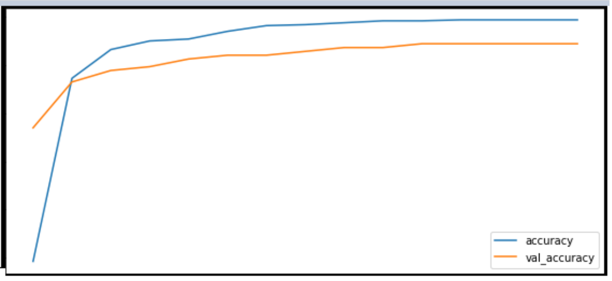
True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	46	0	1	0	5	0	0	0	0	1
1	0	37	0	5	0	1	1	1	4	4
2	2	2	34	5	0	0	3	0	6	1
3	0	1	3	37	0	3	0	1	3	5
4	4	0	0	0	48	0	2	2	0	1
5	2	0	0	1	0	47	6	0	0	0
6	0	1	0	0	3	1	43	0	0	6
7	0	0	0	0	0	0	0	51	0	3
8	0	5	4	6	2	0	0	4	28	3
9	1	0	0	4	1	4	0	0	0	45

```
Classification report for classifier DecisionTreeClassifier():
              precision    recall  f1-score   support

           0       0.84      0.87      0.85        53
           1       0.80      0.70      0.75        53
           2       0.81      0.64      0.72        53
           3       0.64      0.70      0.67        53
           4       0.81      0.84      0.83        57
           5       0.84      0.84      0.84        56
           6       0.78      0.80      0.79        54
           7       0.86      0.94      0.90        54
           8       0.68      0.54      0.60        52
           9       0.65      0.82      0.73        55

    accuracy                           0.77       540
   macro avg       0.77      0.77      0.77       540
weighted avg       0.77      0.77      0.77       540
```

NEURAL NETWORK

We employed a fully connected feedforward neural network. Theoretical study reveals that a single hidden-layer NN may approximate any function (Gardner & Dorling, 1998). Other studies have discovered that NN architectures with more hidden layers occasionally perform better in actual conditions (Koutsoukas et al., 2017). A one-layered network recorded a training accuracy of 99% with a test accuracy of 95% after 15 epochs. The learning curve below shows a good fit of the model.





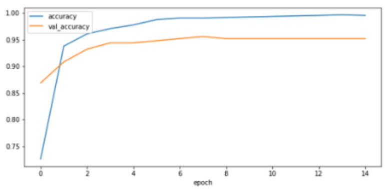
Comparison of performance

CLASSIFIERS	TRAINING ACCURACY	VALIDATION ACCURACY	TIME
Neural network	0.997	0.952	3
Decision Tree	1	0.8	0
Naïve Bayes Classifier	0.877	0.825	0.017
Random Forest	0.847	0.785	0
Support Vector Machine	0.996	0.953	0
KNN	0.991	0.964	0.003

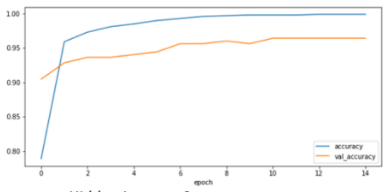
This table shows the training and validation accuracy across the classifiers. It can be deduced that the best model is the neural network which performed best with over 99% training accuracy and 95% validation accuracy. While this looks good, further optimization of the hyperparameter will be done for improved performance.

DEEP NEURAL NETWORK VS SHALLOW NEURAL NETWORK

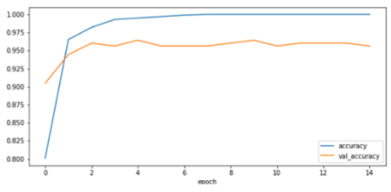
A single hidden layered neural net is a "universal approximator," which means it may estimate any continuous function if the hidden layer has an adequate number of units(Brownlee, 2016). However, as the number of hidden units grows, so does the network's complexity and proclivity to overfit the data (Hamid & Sjarif, 2017). As a result, a bigger hidden layer does not automatically imply a better model, especially when generalizability is a concern.From the graphs and table below , the result does not show significant improvement when using a deep network but there all have a good fit.



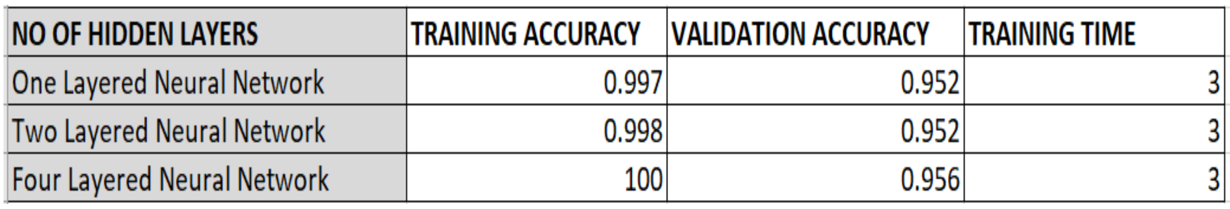
Hidden Layers = 1



Hidden Layers = 2



Hidden Layer = 4



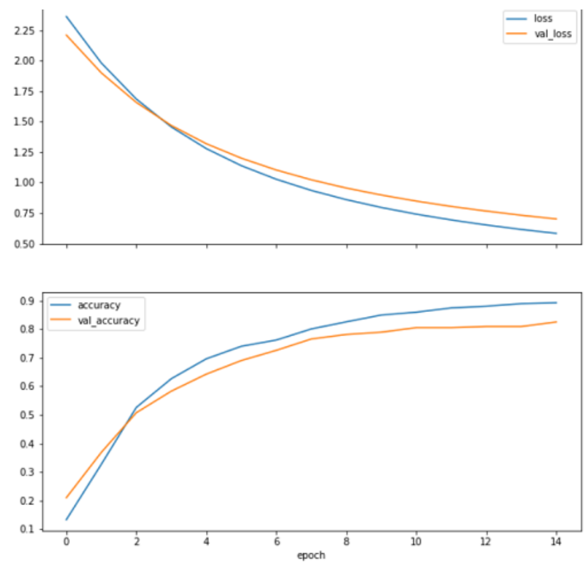
NO OF HIDDEN LAYERS	TRAINING ACCURACY	VALIDATION ACCURACY	TRAINING TIME
One Layered Neural Network	0.997	0.952	3
Two Layered Neural Network	0.998	0.952	3
Four Layered Neural Network	100	0.956	3

What we can now do is make modifications to additional hyper-parameters and, finally, evaluate the performance of the models on the "test dataset," which will tell us whether the training is overfitting and whether our model can make generalizations.

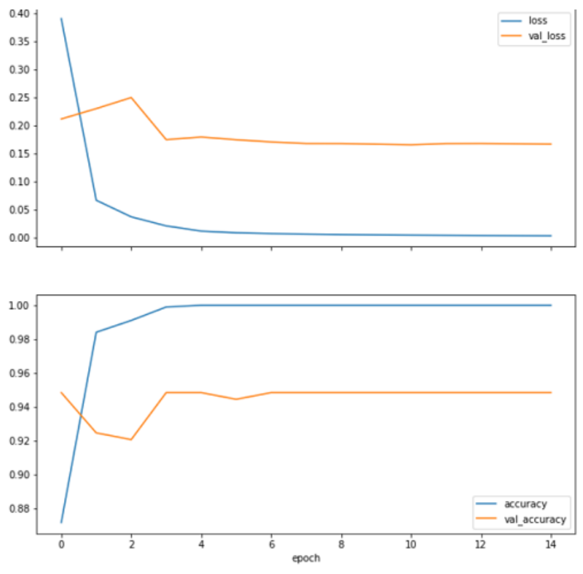
TUNNING THE LEARNING RATE

First we evaluate the training behaviour for a "full dataset batch" (i.e., gradient descent training), while the "learning rate" is varied between the values 0.005 and 1. The weight at the end of each batch is controlled by the Learning rate.

LEARNING RATE	TRAINING ACCURACY	VALIDATION ACCURACY	TRAINING TIME
0.005	0.892	0.825	3
1	100	0.96	3



Lr = 0.005

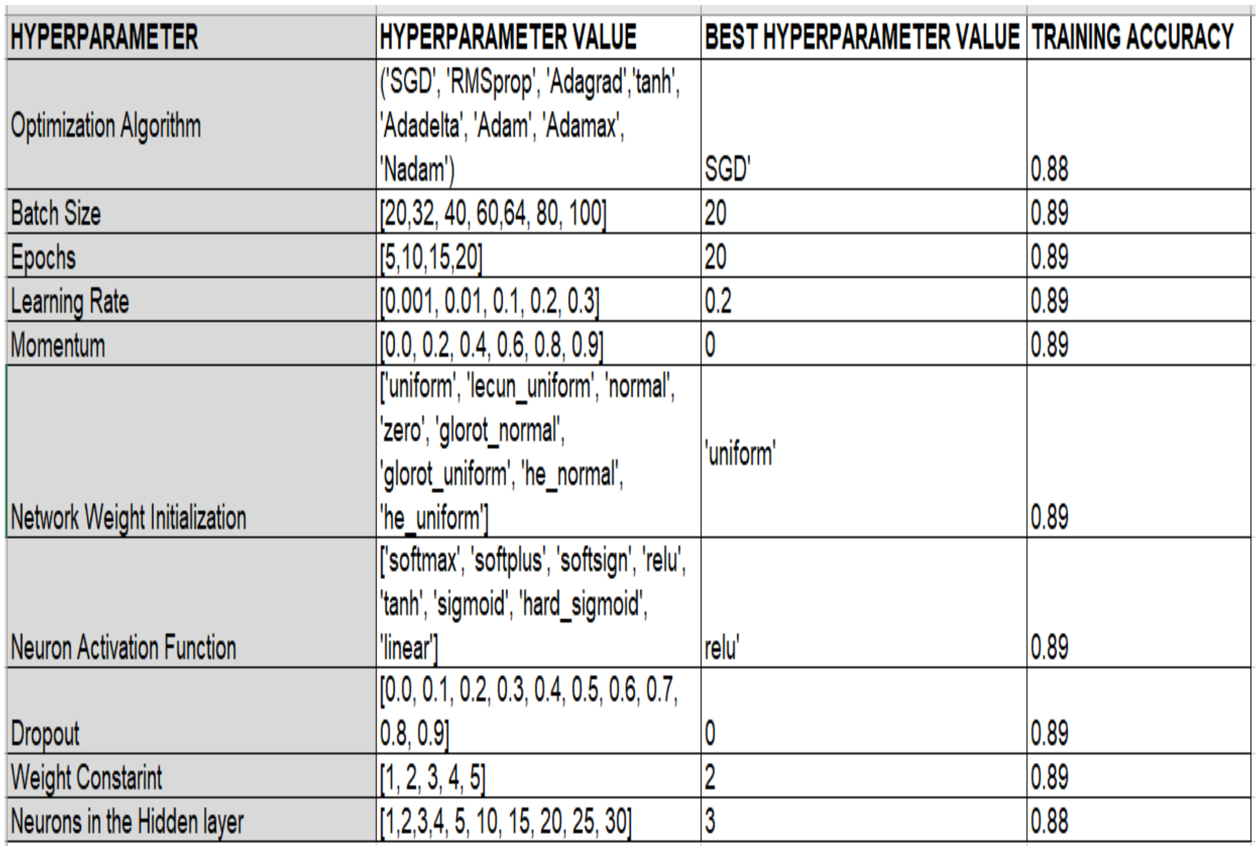


Lr = 1

The results above show that increasing the learning rate from 0.005 to 1 slightly speed up convergence and return a lower final loss on the training set (from 0.773 to 0.375) and increase the training accuracy from 0.892 to 100. The learning rate of 1 is too fast for the model leading to an unstable learning process.

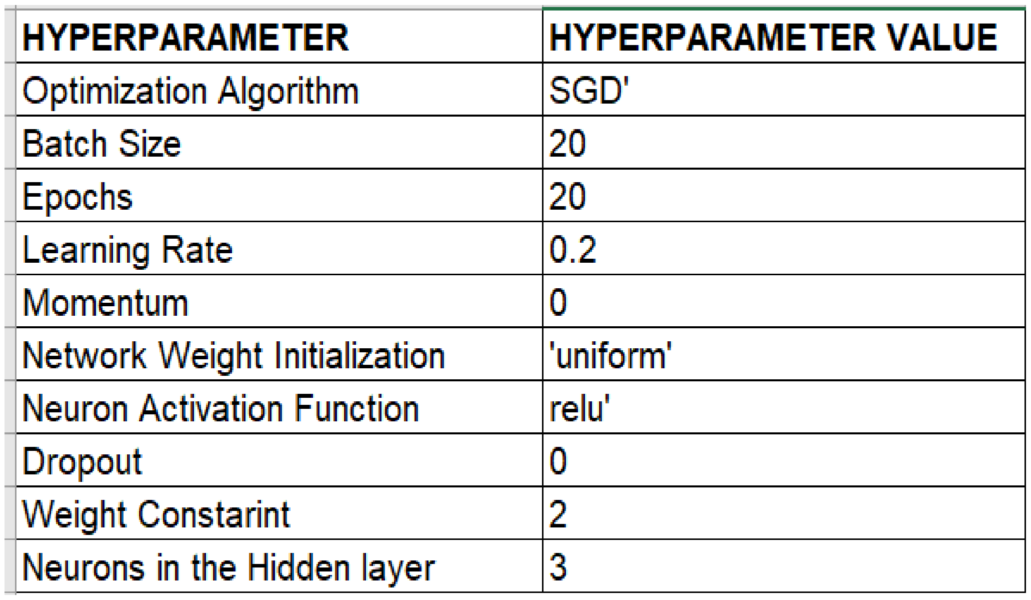
TUNNING THE HYPERPARAMETER WITH GRID SEARCH

In scikit-learn, we wrapped keras models in the KerasClassifier class by creating a function that produces and outputs, and passing the function as the build__fn argument for building the model(Jason, 2020). The KerasClassifier class's function Object accepts default parameters like the learning rate, activation function, and momentum. The experiment demonstrates how cross-validation is used to optimize a classifier using “sklearn.modelselection.GridSearchCV." This module runs "cross-validation" on the training set, automatically dividing it into train and testing data, assessing the scoring metric for each parameter modification. The study compared several examples of hyperparameter tuning, with the best one being picked for final model implementation. The outcome is shown in the table below.



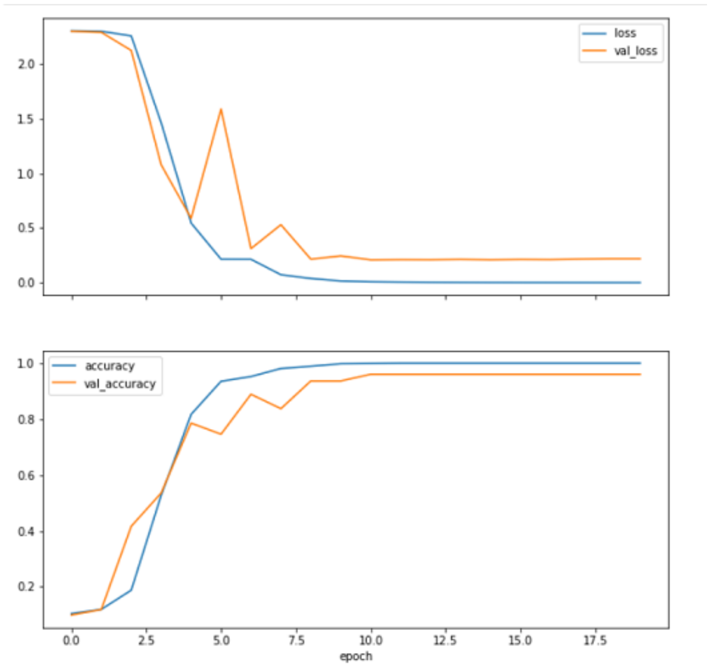
HYPERPARAMETER	HYPERPARAMETER VALUE	BEST HYPERPARAMETER VALUE	TRAINING ACCURACY
Optimization Algorithm	('SGD', 'RMSprop', 'Adagrad','tanh', 'Adadelta', 'Adam', 'Adamax', 'Nadam')	SGD'	0.88
Batch Size	[20,32, 40, 60,64, 80, 100]	20	0.89
Epochs	[5,10,15,20]	20	0.89
Learning Rate	[0.001, 0.01, 0.1, 0.2, 0.3]	0.2	0.89
Momentum	[0.0, 0.2, 0.4, 0.6, 0.8, 0.9]	0	0.89
Network Weight Initialization	['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform']	'uniform'	0.89
Neuron Activation Function	['softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear']	relu'	0.89
Dropout	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]	0	0.89
Weight Constarint	[1, 2, 3, 4, 5]	2	0.89
Neurons in the Hidden layer	[1,2,3,4, 5, 10, 15, 20, 25, 30]	3	0.88

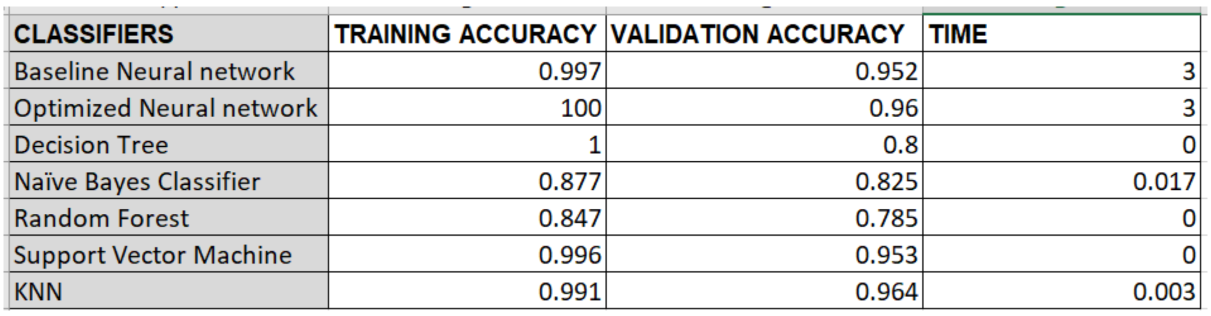
The optimized model was built with the best hyperparameters as shown in the figure below:



HYPERPARAMETER	HYPERPARAMETER VALUE
Optimization Algorithm	SGD'
Batch Size	20
Epochs	20
Learning Rate	0.2
Momentum	0
Network Weight Initialization	'uniform'
Neuron Activation Function	relu'
Dropout	0
Weight Constarint	2
Neurons in the Hidden layer	3

The result, compared with the initial model, showed an incredible improvement. When tuned, feed-forward neural networks achieve good classification performance and beat shallow approaches over a wide range of activity classes. When compared to the other approaches, tuned FNNs outperformed in both training and testing accuracy, as indicated in the table below.





CLASSIFIERS	TRAINING ACCURACY	VALIDATION ACCURACY	TIME
Baseline Neural network	0.997	0.952	3
Optimized Neural network	100	0.96	3
Decision Tree	1	0.8	0
Naïve Bayes Classifier	0.877	0.825	0.017
Random Forest	0.847	0.785	0
Support Vector Machine	0.996	0.953	0
KNN	0.991	0.964	0.003

CONCLUSION

We described the methods and results of improving the FFNN model for identifying handwritten digits in this work. Model hyper-parameter optimization enhanced performance by a significant margin and may thus be regarded a crucial step in building computer vision applications. The analysis demonstrates that by concentrating on the optimization of model's hyperparameter, the outcomes can be improved.

The current study is unique in that it extensively explores most of the parameters of a feedforward neural network architecture which provided an enhanced classification accuracy for a handwritten digit dataset. Deeper models such as the Convolutional neural network may be studied in the future to contrast their performance. Bayesian optimization approach may be used to tune CNN hyperparameters.

REFERENCES

"Dua, Dheeru and Graff, Casey (2017) *{UCI} machine learning repository.* University of California, Irvine, School of Information and Computer Sciences. Available online: http://archive.ics.uci.edu/ml [Accessed 14/04/ 2022].

A. Comeau & C. McDonald. (2019) Analyzing decision trees to understand MNIST misclassification. *- 2019 IEEE MIT Undergraduate Research Technology Conference (URTC).*

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A. & Arshad, H. (2018) State-of-the-art in artificial neural network applications: A survey. *Heliyon,* 4 (11), e00938.

Albahli, S., Alhassan, F., Albattah, W. & Khan, R. U. (2020) Handwritten digit recognition: Hyperparameters-based analysis. *Applied Sciences,* 10 (17), 5988.

Al-behadili, H. N. K. (2016) Classification algorithms for determining handwritten digit. *Iraq J.Electr.Electron.Eng,* 12 (1), 96-102.

Alpaydin, E., Kaynak, C., Alimoglu, F., Al, F. & Glu, I. (2000) Cascading multiple classifiers and representations for optical and pen-based handwritten digit recognition. *of Statistics, University of California, Berkeley, California 94720, USA.* Citeseer.

Bengio, Y., Courville, A. & Vincent, P. (2013) Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 35 (8), 1798-1828.

Bosch, A., Zisserman, A. & Munoz, X. (2007) Image classification using random forests and ferns. *2007 IEEE 11th international conference on computer vision.* Ieee.

Brownlee, J. (2016) How to grid search hyperparameters for deep learning models in python with keras. *Línea].Disponible En: Https://Machinelearningmastery.Com/Grid-Search-Hyperparameters-Deep-Learning-Models-Python-Keras,* .

Chen, F., Chen, N., Mao, H. & Hu, H. (2018) Assessing four neural networks on handwritten digit recognition dataset (MNIST). *arXiv Preprint arXiv:1811.08278,* .

Chugh, R. S., Bhatia, V., Khanna, K. & Bhatia, V. (2020) A comparative analysis of classifiers for image classification. *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence).* IEEE.

Cireşan, D. C., Meier, U., Gambardella, L. M. & Schmidhuber, J. (2010) Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation,* 22 (12), 3207-3220.

Gardner, M. W. & Dorling, S. R. (1998) Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment,* 32 (14-15), 2627-2636.

Hamid, N. A. & Sjarif, N. N. A. (2017) Handwritten recognition using SVM, KNN and neural network. *arXiv Preprint arXiv:1702.00723,* .

Jason, B. (2020) *Use keras deep learning models with scikit-learn in python.* Available online: https://machinelearningmastery.com/use-keras-deep-learning-models-scikit-learn-python/ [Accessed 17/04/ 2022].

Koutsoukas, A., Monaghan, K. J., Li, X. & Huan, J. (2017) Deep-learning: Investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. *Journal of Cheminformatics,* 9 (1), 1-13.

Lejeune, E. (2020) Mechanical MNIST: A benchmark dataset for mechanical metamodels. *Extreme Mechanics Letters,* 36 100659.

Maglogiannis, I. G. (2007) *Emerging artificial intelligence applications in computer engineering: Real word ai systems with applications in ehealth, hci, information retrieval and pervasive technologies* Ios Press.

Savas, B. & Eldén, L. (2007) Handwritten digit classification using higher order singular value decomposition. *Pattern Recognition,* 40 (3), 993-1003.

Y. Lee. (1991) Handwritten digit recognition using *K* nearest-neighbor, radial-basis function, and backpropagation neural networks.

Zhang, M., Jing, W., Lin, J., Fang, N., Wei, W., Woźniak, M. & Damaševičius, R. (2020) NAS-HRIS: Automatic design and architecture search of neural network for semantic segmentation in remote sensing images. *Sensors,* 20 (18), 5292.